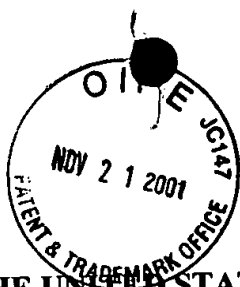


WN-2387



2
BT
12-03-01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of

Takumi Washio

Serial No.: 09/964,749

Group Art Unit: 2185

Filing Date: September 28, 2001

Examiner: Unknown

For: COMPUTER SYSTEM AND METHOD OF CONTROLLING COMPUTATION

Assistant Commissioner of Patents
Washington, D.C. 20231

RECEIVED
NOV 26 2001
Technology Center 2100

SUBMISSION OF PRIORITY DOCUMENT

Sir:

Submitted herewith is a certified copy of Japanese Application Number 2000-299683
filed on September 29, 2000, upon which application the claim for priority is based.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Sean M. McGinn".

Sean M. McGinn

Registration No. 34,386

Date:

11/21/01
McGinn & Gibb, PLLC

Intellectual Property Law

8321 Old Courthouse Road, Suite 200

Vienna, Virginia 22182-3817

(703) 761-4100

Customer No. 21254

日本国特許庁
JAPAN PATENT OFFICE

US

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2000年 9月29日

出願番号

Application Number:

特願2000-299683

出願人

Applicant(s):

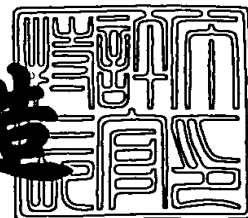
日本電気株式会社

RECEIVED
NOV 26 2001
Technology Center 2100

2001年 6月 5日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2001-3052742

【書類名】 特許願

【整理番号】 35001008

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/16
G06F 15/16

【発明者】

【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

【氏名】 鷲尾 巧

【特許出願人】

【識別番号】 000004237

【氏名又は名称】 日本電気株式会社

【代理人】

【識別番号】 100093595

【弁理士】

【氏名又は名称】 松本 正夫

【手数料の表示】

【予納台帳番号】 057794

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9303563

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 計算機システムとその計算制御方法

【特許請求の範囲】

【請求項 1】 複数のメモリバンクを備える計算機システムにおいて、
計算処理を制御するプロセッサ部からの制御を受けて、指定された演算を前記
プロセッサ部から独立に処理する付帯演算機を、各前記メモリバンク毎に備え、
各前記付帯演算機は、
対応する前記メモリバンク内に記録されたデータに対する演算や、読出し、書
込みを、前記プロセッサ部から送信される命令やデータに基づいて処理すること
を特徴とする計算機システム。

【請求項 2】 前記付帯演算機は、
対応する前記メモリバンク内に記録された、前記プロセッサ部により指定され
たアドレスのデータを読出し、
前記読み出したデータに対して、前記プロセッサ部により指定された演算を実
行し、
演算結果のデータを、前記指定されたアドレスに書き込むことにより、当該ア
ドレスのデータの更新処理を行うことを特徴とする請求項 1 に記載の計算機シ
ステム。

【請求項 3】 前記プロセッサ部により指定されたアドレスのデータを、前
記プロセッサ部により送信されたデータを用いて、前記プロセッサ部により指定
された四則演算を行い、前記指定されたアドレスのデータを演算結果のデータに
更新する手段を備えることを特徴とする請求項 1 又は請求項 2 に記載の計算機シ
ステム。

【請求項 4】 前記プロセッサ部により指定されたアドレスのデータに対し
て、予め設定された値を加算された値に更新する手段、及び予め設定された値を
減算された値に更新する手段を備えることを特徴とする請求項 1 から請求項 3 の
いずれか一つに記載の計算機システム。

【請求項 5】 前記プロセッサ部は、
ベクトル演算による計算処理を実行することを特徴とする請求項 1 から請求項

4 のいずれか一つに記載の計算機システム。

【請求項 6】 前記プロセッサ部は、

複数のプロセッサを備えて、処理対象の計算を各前記プロセッサに割当てて並列処理することを特徴とする請求項 1 から請求項 5 のいずれか一つに記載の計算機システム。

【請求項 7】 複数のメモリバンクを備える計算機システムの計算制御方法において、

計算処理を制御するプロセッサ部が、各前記メモリバンク毎に備えられた、前記プロセッサ部から独立した付帯演算機に対し、演算の実行を指示するステップと、

前記付帯演算機が、

対応する前記メモリバンク内に記録された、前記プロセッサ部により指定されたアドレスのデータを読み出すステップと、

前記読み出したデータに対して、前記プロセッサ部により指定された演算を実行するステップと、

演算結果のデータを、前記指定されたアドレスに書き込むステップを備えることにより、当該アドレスのデータの更新処理を行うことを特徴とする計算制御方法。

【請求項 8】 前記指定されたアドレスのデータを、前記プロセッサ部から送信されたデータを用いて、前記プロセッサ部により指定された四則演算を行い、前記指定されたアドレスのデータを演算結果のデータに更新するステップを備えることを特徴とする請求項 7 に記載の計算制御方法。

【請求項 9】 前記指定されたアドレスのデータに対して、予め設定された値を加算された値に更新するステップ、及び予め設定された値を減算された値に更新するステップを備えることを特徴とする請求項 7 又は請求項 8 に記載の計算制御方法。

【請求項 10】 ベクトル演算による計算処理を実行することを特徴とする請求項 7 から請求項 9 のいずれか一つに記載の計算制御方法。

【請求項 11】 処理対象の計算を、複数のプロセッサに割当てて並列処理

することを特徴とする請求項 7 から請求項 1 0 のいずれか一つに記載の計算制御方法。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、複数のメモリバンクを備える計算機システムに関し、特に、ベクトル計算機や並列計算機等の高速に演算を処理する計算機システムに関する。

【0 0 0 2】

【従来の技術】

従来より、コンピュータの処理を高速化するために、ベクトル演算処理や、並列演算処理等の技術が用いられている。

【0 0 0 3】

スーパーコンピュータ等の大型計算機においては、行列計算や F o r t r a n の D O ループのように、各データに対し同一の手順の演算を繰返し行う処理が多く時間を占めている。ベクトル演算処理は、こうした繰返し行う処理を高速化するために、各行列やデータをそれぞれにベクトルとしてまとめて 1 命令で行列の演算を実行するものであり、行列の各要素毎に 1 つずつ演算を行う必要がないために高速に処理が可能となる。この高速の処理は、各要素に対する命令を、“命令の呼出”、“命令の解読”、“アドレス計算”等々の部分に分けパイプライン制御により同時に並列に実行することにより行われる。

【0 0 0 4】

また、並列処理は、複数のプロセッサやコンピュータを用いて、それぞれに処理を振り分けて並行して実行することにより、処理を高速化するものである。

【0 0 0 5】

また、このようにベクトルにまとめられたり、個々のプロセッサに振り分けられた各データは、途中で各データそれぞれ個別の処理を行う場合も多い。これには、F o r t r a n の D O ループ中に I F 文等の条件付演算を含む場合等がある。

【0 0 0 6】

例えば、ループ内等において、ある配列内の要素を間接参照でメモリからロードして、所定の演算を実行し、その演算結果を同要素が格納されていた元のアドレスにストアするという操作が現れることがあり、これは、有限要素法や分子動力学法の科学技術計算等の多くの場合に現れる。

【0007】

このような演算を含むループにおいては、そのベクトル化（又、並列化）された範囲内においてデータの書込先が重複しない等の保障が必要であり、その保障のない強制的なベクトル化（又、並列化）には、不正な演算結果が出力される危険がある。

【0008】

また同様にベクトル化（又、並列化）が困難な処理の例には、例えば、データ集合内の各データを順次、定められたグループに分類する処理がある。この分類処理のループにおいては、各グループ毎に設けたカウンタ値を当該グループに各データが分類される度に更新するのであり、これの強制的なベクトル化（又、並列化）は、カウンタ値の更新の一貫性が破壊される危険がある。

【0009】

従来では、このようなループのベクトル化（又、並列化）においては、プログラム作成者が、データの格納方法を工夫したり計算順序を変更することにより、ベクトル化（又、並列化）の各単位において更新データの格納先の重複や、カウンタ更新の重複等を回避する必要があった。

【0010】

【発明が解決しようとする課題】

上述したように従来の計算機システムでは、以下に述べるような問題点があった。

【0011】

従来の計算機システムにおいて必要とされていた、更新データの格納先の重複やカウンタ更新の重複等を回避した特別のプログラムを作成することは、プログラマに大きな負担となっていた。

【0012】

更に、このようにプログラムを大幅に変更する必要性は、過去より蓄積されたプログラムを、ベクトル計算機や共有メモリ並列計算機等に移植する際の大きな障害となる。

【 0 0 1 3 】

更に、並列化やベクトル化のための準備作業自身が並列化又はベクトル化できないので、アプリケーションによっては高速化が望めないという問題もある。

【 0 0 1 4 】

本発明の目的は、上記従来技術の欠点を解決し、例えば前述のメモリ内のデータの更新処理や、カウンタの更新を伴うデータの分類処理等の、従来ではベクトル化（又、並列化）が困難であった各種の処理を、容易にベクトル化（又、並列化）して高速に処理することのできる計算機システムと、その計算制御方法を提供することである。

【 0 0 1 5 】

【課題を解決するための手段】

上記目的を達成するため、本発明の計算機システムは、複数のメモリバンクを備える計算機システムにおいて、計算処理を制御するプロセッサ部からの制御を受けて、指定された演算を前記プロセッサ部から独立に処理する付帯演算機を、各前記メモリバンク毎に備え、各前記付帯演算機は、対応する前記メモリバンク内に記録されたデータに対する演算や、読出し、書込みを、前記プロセッサ部から送信される命令やデータに基づいて処理することを特徴とする。

【 0 0 1 6 】

請求項 2 の本発明の計算機システムは、前記付帯演算機により、対応する前記メモリバンク内に記録された、前記プロセッサ部により指定されたアドレスのデータを読出し、前記読み出したデータに対して、前記プロセッサ部により指定された演算を実行し、演算結果のデータを、前記指定されたアドレスに書き込むことにより、当該アドレスのデータの更新処理を行うことを特徴とする。

【 0 0 1 7 】

請求項 3 の本発明の計算機システムは、前記プロセッサ部により指定されたアドレスのデータを、前記プロセッサ部により送信されたデータを用いて、前記プ

ロセッサ部により指定された四則演算を行い、前記指定されたアドレスのデータを演算結果のデータに更新する手段を備えることを特徴とする。

【 0 0 1 8 】

請求項 4 の本発明の計算機システムは、前記プロセッサ部により指定されたアドレスのデータに対して、予め設定された値を加算された値に更新する手段、及び予め設定された値を減算された値に更新する手段を備えることを特徴とする。

【 0 0 1 9 】

請求項 5 の本発明の計算機システムは、前記プロセッサ部において、ベクトル演算による計算処理を実行することを特徴とする。

【 0 0 2 0 】

請求項 6 の本発明の計算機システムは、前記プロセッサ部において、複数のプロセッサを備えて、処理対象の計算を各前記プロセッサに割当てて並列処理することを特徴とする。

【 0 0 2 1 】

請求項 7 の本発明の計算制御方法は、複数のメモリバンクを備える計算機システムの計算制御方法において、計算処理を制御するプロセッサ部が、各前記メモリバンク毎に備えられた、前記プロセッサ部から独立した付帯演算機に対し、演算の実行を指示するステップと、前記付帯演算機が、対応する前記メモリバンク内に記録された、前記プロセッサ部により指定されたアドレスのデータを読み出すステップと、前記読み出したデータに対して、前記プロセッサ部により指定された演算を実行するステップと、演算結果のデータを、前記指定されたアドレスに書き込むステップを備えることにより、当該アドレスのデータの更新処理を行うことを特徴とする。

【 0 0 2 2 】

請求項 8 の本発明の計算制御方法は、前記指定されたアドレスのデータを、前記プロセッサ部から送信されたデータを用いて、前記プロセッサ部により指定された四則演算を行い、前記指定されたアドレスのデータを演算結果のデータに更新するステップを備えることを特徴とする。

【 0 0 2 3 】

請求項 9 の本発明の計算制御方法は、前記指定されたアドレスのデータに対して、予め設定された値を加算された値に更新するステップ、及び予め設定された値を減算された値に更新するステップを備えることを特徴とする。

【 0 0 2 4 】

請求項 1 0 の本発明の計算制御方法は、ベクトル演算による計算処理を実行することを特徴とする。

【 0 0 2 5 】

請求項 1 1 の本発明の計算制御方法は、処理対象の計算を、複数のプロセッサに割当てて並列処理することを特徴とする。

【 0 0 2 6 】

【発明の実施の形態】

以下、本発明の実施の形態について図面を参照して詳細に説明する。

【 0 0 2 7 】

本発明の計算機システムは、複数のメモリバンクのそれぞれに独立に並列に動作する付帯演算機を備えて、各メモリバンク毎に局所的な排他処理を行い、付帯演算機によりメモリバンク内のデータを適切に更新する。これにより、従来ではベクトル化（又、並列化）が困難であった各種の処理に対しても、特別のプログラムを必要とすることなく、正確で高速な演算処理を実現させるのである。

【 0 0 2 8 】

また、このように本発明の計算機システムは、ベクトル計算機や（メモリを共有する）並列計算機等に適応することができる。

【 0 0 2 9 】

図 1 は、本発明の計算機システムをベクトル計算機システムに適応した実施の形態の一実施例の構成を示すブロック図であり、図 2 は、本発明の計算機システムを並列計算機システムに適応した実施の形態の一実施例の構成を示すブロック図である。いずれの形態においても、複数の各メモリバンク 4 0 にそれぞれ付帯演算機 3 0 を接続している。

【 0 0 3 0 】

また、図 1 のベクトル計算機システムの形態においては、ベクトル演算処理を

行うベクトルプロセッサ 1 1 を備え、又、図 2 の並列計算機システムの形態においては、それぞれが並行して演算処理を実行する複数のプロセッサ 1 2 を備えている。また、本発明の計算機システムの、これらプロセッサを備える部分をプロセッサ部と呼ぶものとする。

【 0 0 3 1 】

また、これらプロセッサと各メモリバンク 4 0 との間には、各メモリバンク 4 0 へのアクセスや、各付帯演算機 3 0 への命令やデータの送受を制御するメモリ制御部 2 0 を備えている。

【 0 0 3 2 】

各付帯演算機 3 0 は、対応するメモリバンク 4 0 内に記録されたデータに対して、例えば、アクセス権（排他制御）の設定や解放、データの読出しや書込、データの演算等を行う。また、データの演算においては、メモリバンク 4 0 から読み出したデータと、プロセッサの側から送られたデータとを用いて四則演算等の各種定められた演算を実行する機能をも備える。そして、このメモリバンク 4 0 から読み出したデータのアドレスに、その演算結果のデータを書き込むことにより、メモリバンク 4 0 のデータを更新するのである。また、付帯演算機 3 0 は、このようにメモリバンク 4 0 内のデータを更新するのみに限らず、メモリバンク 4 0 内に新規のデータを書き込む機能等を備えてもよい。

【 0 0 3 3 】

図 3 は、本発明の図 1、図 2 の各実施の形態における付帯演算機 3 0 とメモリバンク 4 0 との接続と、付帯演算機 3 0 の一実施例の構成を示すブロック図である。

【 0 0 3 4 】

図 3 を参照すると、本実施例の付帯演算機 3 0 は、メモリ制御部 2 0 とメモリバンク 4 0 との間に、その双方を結ぶメモリ制御線 7 1、データ線 7 2、アドレス線 7 3 を介して接続している。メモリ制御線 7 1 ではメモリバンク 4 0 を制御する命令を通信し、データ線 7 2 では記録されたデータや記録するデータを送受し、アドレス線 7 3 ではメモリバンク 4 0 内のアクセス先のデータのアドレスを通信する。

【 0 0 3 5 】

また、メモリ制御部 2 0 と付帯演算機 3 0 の間には、メモリアクセスの排他制御を調停するアクセス調停部 6 0 を設けており、メモリ制御部 2 0 とアクセス調停部 6 0 をアクセス調停線 7 4 で、付帯演算機制御部 3 1 とアクセス調停部 6 0 をアクセス調停線 7 5 で接続している。また、メモリ制御部 2 0 から付帯演算機 3 0 に対して、実行する演算の内容や更新処理の内容を通知するための付帯演算機制御線 7 6 を備えている。

【 0 0 3 6 】

次に、本発明の図 1、図 2 の各実施の形態の計算機システムの処理を説明する。

【 0 0 3 7 】

図 4 は、本発明の図 1、図 2 の各実施の形態の計算機システムの処理を説明するための図であり、この例においては、所定の計算により算出される値 “ $f(a, b, c, \dots)$ ” を、メモリバンク 4 0 に記録されたデータ “Y” に加算して更新する処理を説明するものである。なお、“+” の記号は、左辺に示される変数の値に、右辺の値を加算する旨を示す演算子である。

【 0 0 3 8 】

この場合においては、図 1 のベクトルプロセッサ 1 1 及び図 2 のプロセッサ 1 2 は、図 4 の右辺に示されるように更新に用いるデータ “ $f(a, b, c, \dots)$ ” を算出する処理を実行し、その算出結果を付帯演算機 3 0 の側に渡す。そして、付帯演算機 3 0 が、更新対象のデータ “Y” をロードし、このロードしたデータをプロセッサから渡された値と共に指定された演算（図 4 の例では加算）を実行し、この更新された “Y” の値をメモリバンク 4 0 に書き込むのである。

【 0 0 3 9 】

また、付帯演算機 3 0 は、プロセッサの側から独立して演算やデータの読出し書き込みを実行する演算機であり、プロセッサから受けた命令やデータはデータ用シフトレジスタ 3 3 やアドレス用シフトレジスタ 3 4 に順次記録され順番に実行する。このため、プロセッサの側においては、ベクトル演算処理のループ中等においてメモリバンク 4 0 のデータを更新する必要が発生した場合には、その更

新処理を付帯演算機 3 0 に対して命令することにより、処理を中断することなく要求される演算を適切に実行することができる。

【 0 0 4 0 】

このため、ループ中にデータの更新処理が発生する場合等においても、従来では必要とされた特別のプログラムの作成等の処理が不要となり、利用者は高速の演算処理を簡易に実行することができる。

【 0 0 4 1 】

次に、上述の本発明の計算機システムの処理をフローチャートを参照してより詳細に説明する。図 5 は、本発明の図 1、図 2 の各実施の形態の計算機システムの処理を説明するためのフローチャートである。

【 0 0 4 2 】

まず、図 1 のベクトルプロセッサ 1 1 及び図 2 のプロセッサ 1 2 は、更新に用いるデータを算出する（ステップ 5 0 1）。この更新に用いるデータとは、図 4 の例においては右辺の “ $f(a, b, c, \dots)$ ” であり、プロセッサはこれを算出するのである。

【 0 0 4 3 】

次に、プロセッサは、この算出結果である更新に用いるデータと、更新対象のデータのアドレスの指定と、更新における演算の内容を、メモリ制御部 2 0 に対し通知する（ステップ 5 0 2）。図 4 の例においては、更新対象のデータのアドレスは、変数 “Y” のアドレスであり、更新における演算の内容は、当該変数 “Y” に対して更新に用いるデータを加算することである。このようにプロセッサは、付帯演算機 3 0 に実行させる加算処理等の四則演算やその他の演算処理を指定して、メモリ制御部 2 0 に通知するのである。

【 0 0 4 4 】

メモリ制御部 2 0 は、プロセッサからのこの通知を受けると、付帯演算機 3 0 に対してデータの更新処理を要求する（ステップ 5 0 3）。この要求先の付帯演算機 3 0 は、更新対象のデータを記録するメモリバンク 4 0 に対して設置されているものである。

【 0 0 4 5 】

ここで、メモリ制御部 2 0 は、付帯演算機 3 0 に対して、更新に用いるデータをデータ線 7 2 から、更新されるデータのアドレスをアドレス線 7 3 から、実行する演算の内容の指定を付帯演算機制御線 7 6 から送信することにより、更新処理を要求する。

【 0 0 4 6 】

付帯演算機制御部 3 1 は、メモリ制御部 2 0 から送信される信号を検知し、データ用シフトレジスタ 3 3 に更新に用いるデータを記録し、アドレス用シフトレジスタ 3 4 に更新対象のデータのアドレスを記録する。そして、この格納が終了すると、メモリ制御部 2 0 に対して格納動作の終了の旨を付帯演算機制御線 7 6 を通して通知する。

【 0 0 4 7 】

付帯演算機 3 0 は、データ用シフトレジスタ 3 3 及びアドレス用シフトレジスタ 3 4 に記録されたデータやアドレスを、記録された順番に読み出して演算を実行する。

【 0 0 4 8 】

ここで、メモリ制御部 2 0 から受けたデータを実行する順番が到来した場合には、付帯演算機 3 0 は、そのメモリアクセスの処理を実行する。データ用シフトレジスタ 3 3 の出口にデータが現れた場合には、付帯演算機制御部 3 1 は、データ用シフトレジスタ 3 3 及びアドレス用シフトレジスタ 3 4 の出口のデータを取り出して、それぞれをレジスタ（レジスタ 3 7 とレジスタ 3 5）に格納し、アクセス調停部 6 0 に対してメモリバンク 4 0 のアクセス権をアクセス調停線 7 5 を通して要求する（ステップ 5 0 4）。

【 0 0 4 9 】

アクセス調停部 6 0 は、前記アクセス権の要求を受けると、メモリバンク 4 0 の接続をメモリ制御部 2 0 から付帯演算機 3 0 の側に切り替える。例えば、図 3 の装置構成において、スイッチ 8 1、8 2、8 3、8 4 をオフとし、スイッチ 8 5、8 6 をオンにするのである。そして、付帯演算機制御部 3 1 に対して、メモリバンク 4 0 へのアクセス許可をアクセス調停線 7 5 を通して発行する。

【 0 0 5 0 】

付帯演算機制御部 3 1 は、メモリバンク 4 0 のデータへのアクセス権を得ると、更新対象のデータをレジスタ 3 6 に読み込む（ステップ 5 0 5）。

【 0 0 5 1 】

そして、付帯演算機演算部 3 2 は、このレジスタ 3 6 に記録された更新対象のデータと、レジスタ 3 7 に記録された更新に用いるデータを用いて、指定された演算を実行し、その演算結果をレジスタ 3 8 に格納する（ステップ 5 0 6）。

【 0 0 5 2 】

そして、付帯演算機制御部 3 1 は、スイッチ 3 2 - 1 をオンにし、更新されたデータをメモリバンクに書き込む（ステップ 5 0 7）。

【 0 0 5 3 】

付帯演算機制御部 3 1 は、前記書き込み処理が終了すると、アクセス調停部 6 0 に対して、メモリバンクのアクセス権の返還をアクセス調停線 7 5 を通して通知する（ステップ 5 0 8）。アクセス調停部 6 0 は、アクセス権の返還の通知を受けると、ステップ 5 0 4 において変更したスイッチの状態を元に戻す等により、メモリバンク 4 0 の接続を付帯演算機 3 0 からメモリ制御部 2 0 の側に切り替えて元の状態に戻すのである。

【 0 0 5 4 】

以上により、付帯演算機 3 0 を用いるデータの更新処理の一連の動作が終了する。

【 0 0 5 5 】

付帯演算機 3 0 にシフトレジスタ 3 3、3 4 を設けた理由は、同一メモリバンクへのデータ更新要求が短い時間間隔で生じた場合でも、更新要求を貯めておき、メモリ制御部 2 0 が速やか次の処理に移れるようにするためである。

【 0 0 5 6 】

また、付帯演算機 3 0 によるデータ更新中においては、例えば上記の例においては、スイッチ 8 1、8 2、8 3、8 4 がオフになっているために、メモリ制御部 2 0 は、メモリバンク 4 0 にアクセスすることはできないが、付帯演算機 3 0 に対して、更新に用いるデータと更新されるデータのアドレスをそのシフトレジスタ 3 3、3 4 に書き込むことができる。

【 0 0 5 7 】

また、図 5 のフローチャートの例においては、プロセッサの側から更新に用いるデータを送信するものであったが、他に例えば、記録されたデータの値を“1” つづつ増加させるカウンタの処理の場合等においては、そのカウンタ処理の旨と、更新対象のデータのアドレスを通知するのみで十分であり、更新に用いるデータの値を送る必要はない。本発明の計算機システムは、図 5 のフローチャートと同様にしてこうした処理に対応させることができる。

【 0 0 5 8 】

図 6 は、本発明の図 1、図 2 の各実施の形態の計算機システムによるカウンタの加算処理を説明するためのフローチャートである。付帯演算機 3 0 は、カウンタを排他的に加算更新を実行する。

【 0 0 5 9 】

まず、図 1 のベクトルプロセッサ 1 1 又は図 2 のプロセッサ 1 2 は、メモリ制御部 2 0 に対して、加算更新の対象である整数データのアドレスと、当該整数データに対して加算更新を実行する旨を通知する（ステップ 6 0 1）。

【 0 0 6 0 】

メモリ制御部 2 0 は、これを受けると、アクセス調停部 6 0 に対して加算演算の許可をアクセス調停線 7 4 を通して要求する（ステップ 6 0 2）。アクセス調停部は、もし付帯演算機 3 0 の側にアクセス権を与えていない場合には、メモリ制御部 2 0 にアクセス権を与えてその旨をアクセス調停線 7 4 を通して通知し、又、オン・オフの変更の必要なスイッチを変更する。

【 0 0 6 1 】

メモリ制御部 2 0 は、アクセス権を得ると付帯演算機制御部 3 1 に対して、加算更新の実行を付帯演算機制御線 7 6 を通して要求する（ステップ 6 0 3）。付帯演算機制御部 3 1 は、これを受けてスイッチ 3 4 - 1 をオンにする。

【 0 0 6 2 】

次に、メモリ制御部 2 0 は、加算更新対象のデータを読み出す（ステップ 6 0 4）。これは、加算更新対象のデータのアドレスをアドレス線 7 3 に出力して、その加算更新対象のデータをデータ線 7 2 から受け取ることができる。また、付

帯演算機制御部 31 においても、ここで加算対象のデータ及びそのデータのアドレスのそれぞれを、前記読み込み動作中にデータ線 77 とアドレス線 73 のそれぞれから読み取って、それぞれをレジスタ（レジスタ 36 とレジスタ 35）に格納することができる。

【0063】

アクセス調停部 60 は、このメモリ制御部 20 による動作が終了すると、アクセス権を自動的に付帯演算機 30 の側に切り替える（ステップ 605）。例えば、スイッチ 81、82、83、84 をオフにするのである。

【0064】

付帯演算機制御部 31 は、付帯演算機演算部 32 を用いて加算対象のデータの加算の演算を実行する（ステップ 606）。これは、付帯演算機制御部 31 が、加算対象のデータをレジスタ 36 から読み出して、所定の加算演算を実行し、更新されたデータをレジスタ 38 に格納するのである。

【0065】

そして、付帯演算機制御部 31 は、ここで更新されたデータを加算対象のデータの元のアドレスに書き込み、そのデータを更新する（ステップ 607）。これは、付帯演算機制御部 31 が、スイッチ 86 と 32-1 とをオンにし、レジスタ 38 に格納されている更新されたデータを、レジスタ 35 に示される加算対象のデータのメモリバンク 40 内の元のアドレスに書き込むのである。

【0066】

付帯演算機制御部 31 は、この書込処理が終了すると、アクセス調停部 60 に対してアクセス権の返還を通知する（ステップ 608）。アクセス調停部 60 は、アクセス権の返還の通知を受けると、各スイッチを所定の状態に戻す等により、メモリバンク 40 の接続を付帯演算機 30 からメモリ制御部 20 の側に切り替えて元の状態に戻すのである。

【0067】

以上により、付帯演算機 30 を用いるカウンタの更新処理の一連の動作が終了する。

【0068】

このように、データの加算更新中においては、他のメモリアクセスを許さないようにすることにより、カウンタの値を正確に一貫性を保って更新することができる。

【 0 0 6 9 】

次に、本発明の図 1 や図 2 の各実施の形態を、具体的な処理の実施例を用いて説明する。

【 0 0 7 0 】

まず第 1 の実施例として、疎行列の演算を考える。疎行列とは、行列成分に“0”以外の要素が少ない行列である。このため、疎行列の演算においては、値が“0”である殆どの要素に対して演算を実行する必要がない等の場合が多く、その演算に際しては様々な高速処理の技術が用いられている。

【 0 0 7 1 】

図 7 は、本実施例において演算する式を示す図であり、ここでは疎行列 A に対して配列 x を掛けて得られる配列を、配列 y に対して加算する処理を演算するのである。つまり、“ $y = y + A x$ ”（又“ $y += A x$ ”）の式に示される演算を実行するのである。

【 0 0 7 2 】

図 8 は、こうした“ $y = y + A x$ ”の行列演算を処理するアルゴリズムの一例を示すフローチャートである。

【 0 0 7 3 】

図 8 においては、疎行列 A の“0”以外の要素（非ゼロ成分）の総数を“n”と示し、その非ゼロ成分の各値を“配列 a”の各要素が示すものとする。つまり、配列 a の要素である n 個の値、 $a(i)$ ($i = 1 \sim n$) が、疎行列 A の非ゼロ成分の各値を示すのである。

【 0 0 7 4 】

また、配列 a が示す疎行列 A の要素の座標を示す行番号と列番号とを、配列 row と配列 column とのそれぞれにより示すものとする。つまり、各 i ($= 1 \sim n$) における $a(i)$ の値は、疎行列 A の ($row(i)$ 、 $column(i)$) の座標の値である。また、配列 a が、疎行列 A の各非ゼロ成分を示す順番

は任意である。

【0075】

図8においては、このように予め疎行列Aの非ゼロ成分を抽出して、演算処理をその非ゼロ成分に対してのみ行うことにより、必要とする演算回数を大幅に削減し高速化を図っているのである。この技術は、従来よりベクトル計算機等の高速処理を行う計算機システムにおいて実施されている。

【0076】

図9と図10は、この図8に示されるアルゴリズムを用いた、図7の例の“ $y = y + Ax$ ”の行列演算を示す図である。

【0077】

図7の例においては、行列Aの非ゼロ成分は4個であり（よって $n=4$ ）、これをここでは、座標（1、1）（1、2）（3、1）（5、5）の順番に配列aに格納している。よって $a(i)$ （ $i=1\sim 4$ ）の値は、順に“5”、“6”、“7”、“8”である。

【0078】

そして、この各配列a、column、row、x、yを用いて演算を行うことができ、プロセッサは、更新に用いるデータである各 $k=1\sim 4$ に対する“ $a(k) * x(column(k))$ ”の値を図9に示されるように算出し、更新対象のデータのアドレスである各 $k=1\sim 4$ に対する“ $y(row(k))$ ”のデータのアドレスを検出し、これらを加算更新を行う旨の命令と共にメモリ制御部20に送る。そして、付加演算機30が、メモリ制御部20からこれらのデータや命令を受け付けて、配列yの更新を実行する。

【0079】

よって、yの各要素が $y(1)=19$ 、 $y(3)=17$ 、 $y(5)=14$ として更新され、yは、 $y=(19, 8, 17, 10, 14)$ に更新される。

【0080】

ここで注意する点は、行列Aにおいては、1つの行に非ゼロ成分が座標（1、1）と座標（1、2）との2箇所が存在することである。“ $y(row(k))$ ”に対して“ $a(k) * x(column(k))$ ”の値を加算し更新する処理

は、この座標（１、１）と座標（１、２）とのそれぞれを順次実行する必要があるものであり、この双方の更新を同時に処理したのでは不正な結果が出力されることとなる。

【0081】

例えば、更新前の $y(1)$ の値は“7”であり、この $y(1)$ の値はインデックス $k=1$ と $k=2$ の時に更新される。ここで、図10においては、インデックス順にまず $k=1$ の更新を行い“12”を加えた後に、続いて $k=1$ の更新を行い“0”を加えるため、 $y(1)$ は正しく“19”に更新されている。しかし、ここでの $k=1$ と $k=2$ の更新が同時に実行されると、同じ $y(1)$ の値がそれぞれに“19”と“7”に更新されることになり、不正な結果が出力される危険がある。

【0082】

この図8に示される演算処理は、疎行列 A の1つの行に2以上の非ゼロ成分が存在する可能性があるために、もしベクトル計算機や共有メモリ並列計算機により処理しようとしても、ステップ803において配列 y の同一の要素を更新する命令が、ベクトル化された1つの単位内に発生したり、異なるプロセッサに並列化されて更新が同時に要求されることにより、正確な更新ができないという危険がある。このため、通常のベクトル計算機や共有メモリ並列計算機では、インデックス k に関してベクトル化又並列化を行うことはできなかった。しかし、図3に示した付帯演算機30を以下のように用いることにより、インデックス k に関してベクトル化及び並列化が可能になる。

【0083】

図11は、本発明の図1のベクトル計算機システムの形態における行列演算“ $y = y + Ax$ ”の処理を説明するためのフローチャートである。

【0084】

図11においては、図8の説明において示されたものと同様に、（疎）行列 A の非ゼロ成分（全 n 個）を任意の順番により、配列 $a(i)$ ($i=1 \sim n$) により示し、その各 $a(i)$ が示す行列 A の要素の座標が、配列 row と配列 $column$ を用いて ($row(i)$ 、 $column(i)$) と示されるものとする。

【0085】

また、図11においては、インデックス k ($=1 \sim n$) を用いて、これをベクトルレジスタ長 VL 毎に区切ってベクトル処理を実施するものである。このベクトルレジスタ長 VL は、1回のベクトル化においてまとめられる要素の数を示すものであり、個々のベクトル計算機において設定された所定の値が使用される。また、これによりまとめられる最後のセグメントは、 VL よりも短くなる可能性があるため " $L = \min(n - k + 1, VL)$ " によりセグメントの長さを設定することで、配列 a 等における未定義のデータ (番号 n を超過する配列のデータ) を読み出す等の問題を解消している。

【0086】

また、計算処理を実行するステップ903においては、ベクトルレジスタ長 VL にまとめられたベクトルレジスタとして、 $Vcol$ 、 $Vrow$ 、 Va 、 Vax を定義して用いている。

【0087】

図11を参照すると、本実施の形態のベクトル計算機システムでは、行列演算 " $y = y + Ax$ " を以下のように行っている。

【0088】

まず始めに、インデックス k に初期値である "1" を設定し (ステップ901)、この k の値が非ゼロ成分の総数 " n " を超えるまでステップ903の計算処理を実行する (ステップ902)。

【0089】

ステップ903の計算処理においては、まず $1 \sim n$ までを、 VL の倍数毎に区分けしてベクトル化する範囲を設定している。つまり、 $(1 \sim VL)$ 、 $(VL + 1 \sim 2 * VL)$ 、 $(2 * VL + 1 \sim 3 * VL)$ 、…のように " n " を超えるまでを区分けしている。

【0090】

ここでは、まず、 i_vector_load 命令により、アドレス用ベクトルレジスタ $Vcol$ に、整数配列 $column$ のインデックス k から始まる L 個の連続にならんだデータをロードする。また、 k の値は、ステップ904の更新

処理において示されるようにVLの整数倍を成している。同様に、アドレス用ベクトルレジスタVrowに、整数配列rowのデータをロードする。

【0091】

また、r__vector__load命令により、ベクトルレジスタVaに、実数配列aから非ゼロ成分の実数データをロードする。

【0092】

次に、gather命令により、ベクトルレジスタVxに、実数配列xのデータにおけるアドレス用ベクトルレジスタVcolに順次示される位置のデータをロードする。

【0093】

その後、vector__op命令において、ベクトル演算パイプラインを用いVaとVxのデータの積を計算し、ベクトルレジスタVaxに格納する。このベクトル演算は各ベクトルレジスタの要素毎に積を算出するものであり、つまり、ここで“ $Vax = Va * Vx$ ”の演算は、“ $Vax(i) = Va(i) * Vx(i)$ ” ($i = 1 \sim L$) をベクトル演算パイプラインにより演算することを意味する。

【0094】

最後にscatter&add命令により、実数配列yにおけるアドレス用ベクトルレジスタVrowに順次示される位置のデータに、付帯演算機30によりVaxのデータを加算し、yをもとの場所に格納してデータを更新する。

【0095】

ここで、scatter&add命令は、図3の付帯演算機30を用いてscatter処理により分配されたデータの加算更新を行うことを指示するコマンドである。すなわち、ベクトルプロセッサは、ベクトルレジスタVaxの各データを、配列y内の対応するデータを格納すべきメモリバンクに、アドレス用ベクトルレジスタVrowを参照してscatter処理により分配し、付帯演算機30が分配されたデータを用いてy内の対応するデータの加算更新を実行するのである。

【0096】

そして、このステップ 9 0 3 の VL 個の要素を一まとめにしたベクトル演算が終了すると、インデックス k に VL を加算して更新し（ステップ 9 0 4）、再びステップ 9 0 2 以下の処理に戻り計算を続行する。

【0 0 9 7】

そして、行列 A の全ての非ゼロ成分の演算処理が終了すると、ベクトルプロセッサと付帯演算機 3 0 の間で同期を取り（ステップ 9 0 5）、付帯演算機 3 0 に加えられるデータが残っていないかどうかの確認等を行い更新動作を終了する。

【0 0 9 8】

以上のように、本実施の形態のベクトル計算機システムを用いることにより、容易にインデックス k に関してベクトル化を行うことができる。例えば、前述の図 7 の式の計算においては、 $y(1)$ の値の $k = 1, 2$ における更新は、ベクトルプロセッサ 1 1 から順次付帯演算機 3 0 に対して更新の命令が送信され、これが順次シフトレジスタに記録されて順番に実行されるため、正しく更新が実行されるのである。

【0 0 9 9】

図 1 2 は、本発明の図 2 の共有メモリ並列計算機システムの形態における行列演算 “ $y = y + Ax$ ” の処理を説明するためのフローチャートである。図 1 2 では、1 から n までのインデックスを 2 つに分割して、それぞれを 2 つのプロセッサ 1 2 により並列に処理している。また、ここでフローチャート中の “ $[n/2]$ ” の記号は、 $n/2$ よりも小さい整数の中で最大の値を示すものとする。

【0 1 0 0】

先のベクトル計算機システムにおいては、同一のデータの更新処理がベクトル化された 1 つの単位内に発生した場合に、不正な更新が行われることを防止するものであった。ここでの、共有メモリ並列計算機システムの形態においては、同一のデータの更新処理が異なるプロセッサに割当てられた場合に、それぞれのプロセッサにより個別に更新が実行され、不正な更新が行われることを防止する。

【0 1 0 1】

ここで、ステップ 1 2 0 4 とステップ 1 2 0 8 における計算処理においては、先に説明したベクトル計算機の場合と同様に、演算子 “ $+$ ” に示される右側の

被演算データに左側の被演算データを加えて更新する処理を、図3の付帯演算機30を用いて処理するのである。

【0102】

すなわち図2の各プロセッサ12は、 $a(k) * x(\text{column}(k))$ の値を計算し、その計算結果のデータを $y(\text{row}(k))$ のアドレスの指定と加算処理を実行する旨の命令と共に、メモリ制御部20に送る。

【0103】

メモリ制御部20は、この $y(\text{row}(k))$ のデータの格納された付帯演算機30に対して、これらのデータやアドレスと共に、加算処理を実行する旨の命令を送る。付帯演算機30は、この命令を受けて $y(\text{row}(k))$ の加算更新を実行するのである。

【0104】

前記のベクトル計算機システム及び共有メモリ並列計算機システムでの実施例のように、メモリバンクに接続された付帯演算機でデータの加算更新を行うことにより、最終的な加算結果の正当性が守られる。

【0105】

この際、各メモリバンクでの加算更新は逐次に行われるが、それぞれメモリバンクが他とは独立に動作するので全体として高速な演算が実現される。

【0106】

続いて、第2の実施例として、カウンタの更新処理を考える。第1の実施例と同様に、同一のカウンタの更新がベクトル化された1つの単位内に発生した場合や、並列処理における異なるプロセッサから同時に発生した場合においても、カウンタを正確にかつ高速に更新するのである。

【0107】

この実施例におけるカウンタの更新処理は、複数個の実数値のデータをその整数部分が等しいものの毎にそれぞれを分類する分類処理において、各グループに分類されたデータをカウントする（又そのデータの位置を記録する）ものである。

【0108】

図13は、本実施例における実数値データの整数部分に基づく分類処理の内容を説明するための図である。分類対象のデータとして5つの実数値を備える配列 $x = (1.2, 2.0, 1.4, 4.5, 2.5)$ がある。この x の各要素のデータを整数部分に基づいて分類し、分類結果を順次配列 $list(j, i)$ に記録するのである。ここで、配列 $list(j, i)$ は、各列 (i) において分類された整数部分の値を示し、各行 (j) において分類されたデータの数を示し、この配列の (j, i) 成分は、整数部分の値が “ i ” である “ j ” 番目に分類されたデータのインデックスを示すものである。

【0109】

図14は、こうした実数データの整数部分に基づく分類を処理するアルゴリズムの一例を示すフローチャートである。

【0110】

図14においては、分類する実数データの総数を “ n ” とし、その各実数データを配列 x により $x(i)$ ($i = 1 \sim n$) と示す。また関数 “ int ” は実数データの整数部分を出力する。

【0111】

また、配列 $count$ により、各整数グループに分類された実数データの総数を示す。例えば、“ $count(2)$ ” により、これまでに分類された整数部分が “2” である実数データの総数が示されるのである。また、前述の様に2次元配列 $list(j, i)$ は、分類されたデータの元の配列 x におけるインデックスを示す。ここで、配列 $count$ や2次元配列 $list$ のサイズは、予め適切なサイズを設定しておく。また、配列 $count$ の各要素の値は、計算処理の実行前に予め “0” に初期化しておく。また、“ $count(i)++$ ” は、配列の要素 “ $count(i)$ ” の値を1つ増加させる命令を意味する。

【0112】

図14のステップ1403における計算処理では、各インデックス k において、 $x(k)$ の整数部分 “ i ” を計算し、その整数部分 “ i ” に該当するカウンタ値 “ $count(i)$ ” の値を “1” 増加させる。そして、2次元配列 $list$ における、当該整数部分 “ i ” の当該カウンタ値 “ $count(i)$ ” を示す要素

“list (count (i), i)” に対して、現在のインデックスの値 “k” を代入する。この操作を全ての k (= 1 ~ n) に対し順次実行することにより、図 1 3 の例に示されるように実数データが分類されるのである。

【0 1 1 3】

図 1 5 は、この図 1 4 に示されるアルゴリズムを用いた、図 1 3 の例の実数データ $x = (1.2, 2.0, 1.4, 4.5, 2.5)$ を分類する処理を説明するための図である。

【0 1 1 4】

図 1 5 においては、配列 x の次数データを順次 1 つずつ図 1 4 のアルゴリズムに基づいて、各グループ毎にカウントすることにより、インデックスの値 “k” の代入先である 2 元配列 list の座標が求められている。

【0 1 1 5】

しかし、図 1 4 に示される分類処理では、整数部分が等しいデータを分類する処理が 1 つのベクトル化の単位内に発生したり、異なるプロセッサから同時に発生する可能性があり、配列 count や 2 元配列 list が不正に更新される危険があるため、従来のベクトル計算機や共有メモリ並列計算機ではインデックス k に関してベクトル化及び並列化ができなかった。

【0 1 1 6】

しかし、図 3 に示した付帯演算機 3 0 を以下のように用いることにより、インデックス k に関するベクトル化及び並列化が可能になる。

【0 1 1 7】

図 1 6 は、本発明の図 1 のベクトル計算機システムの形態における実数データの整数部分に基づく分類処理を説明するためのフローチャートである。

【0 1 1 8】

図 1 6 においては、図 1 4 の説明において示されたものと同様に、分類対象の実数データ（全 n 個）を、配列 $x(i)$ ($i = 1 \sim n$) により示し、その各要素 $x(i)$ の分類を、前述の様に配列 count を用いて、2 元配列 list に記録するのである。

【0 1 1 9】

また、図11においては、第1の実施例の場合と同様にインデックス k ($=1 \sim n$) を用いて、これをベクトルレジスタ長 VL 毎に区切ってベクトル処理を実施するものである。また、これによりまとめられる最後のセグメントは、 VL よりも短くなる可能性があるため " $L = \min(n - k + 1, VL)$ " により適切なセグメントの長さを設定する。

【0120】

また、計算処理を実行するステップ1603においては、ベクトルレジスタ長 VL にまとめられたベクトルレジスタとして、 Vx 、 $Vint$ 、 Vcn 、 Vad を定義して用いている。

【0121】

この計算処理においては、2元配列 $list$ の各要素 $list(j, i)$ を、1次元配列に $list(m * (i-1) + j) = list(j, i)$ の方式により変換して指定している。また、ここで " m " は、2元配列配列 $list(j, i)$ の第1次元目の総個数である（つまり、2元配列配列 $list$ は、各 i において $list(1, i)$ から $list(m, i)$ までの要素を備える。）。

【0122】

また、図16のフローチャートの処理における計算処理を実行するステップ1603以外は、図11に示される第1の実施例と同様である。

【0123】

図16の実施例の計算処理においては、まず、`r_vector_load`命令により、実数配列 x 内のインデックス k から $k+L-1$ までの各データを、ベクトルレジスタ Vx にロードする。

【0124】

次に、`vector_op`命令では、ベクトル演算パイプラインを用いて Vx 内の各実数データの整数部を計算し、アドレス用ベクトルレジスタ $Vint$ に格納する。つまり、 $Vint(i) = \text{int}(Vx(i))$ 、($i=1 \sim L$)の演算を行う。

【0125】

次に、`gather&inc`命令により、各整数部分 $Vint(i)$ の値の力

カウンタ値 $\text{count}(\text{Vint}(i))$ の値を “1” 増加させて、 gather 処理によりベクトルレジスタ Vcn にロードする。ここで、式の右辺 “ $\text{count}(\text{Vint}(i))$ ” の項の左側につけられた演算子 “++” は、右辺の値を左辺に代入する前に、その “ $\text{count}(\text{Vint}(i))$ ” の項の値を “1” 増加させる命令を意味する。

【0126】

ここで、 gather\&inc 命令においては、単に gather 処理によるデータのロードを行うだけでなく、ロードされる整数データの排他的な加算更新を各メモリバンク 40 において付帯演算機 30 により行うことを指示する。

【0127】

次に、 vector_op 命令においては、ベクトル演算パイプラインを用いて、各要素の整数部分の値 $\text{Vint}(i)$ とカウンタ値 $\text{Vcn}(i)$ を基に配列 list の格納先インデックスを計算し、アドレス用ベクトルレジスタ Vad に格納する。

【0128】

最後に、 scatter 命令においては、この Vad の格納先インデックスの値を参照しながら、 list 内の対応するアドレスに各インデックスの値 ($k \sim k+L-1$) を分配する。

【0129】

以上のように、本実施の形態のベクトル計算機システムを用いることにより、容易にインデックス k に関してベクトル化を行うことができる。例えば、前述の図 13、図 15 に示される分類処理の例においては、 $k=1, 3$ において共通の整数部分 “1” が示され、 $k=2, 5$ において共通の整数部分 “2” が示されているため、従来のベクトル計算機システムではベクトル化することができなかった。しかし、本実施の形態のベクトル計算機システムでは、これらを全て 1 つのベクトルにベクトル化した場合においても、ベクトルプロセッサ 11 から順次付帯演算機 30 に対して更新の命令が送信され、これが順次シフトレジスタに記録されて順番に実行されるため、正しく更新が実行されるのである。

【0130】

図 1 7 は、本発明の図 2 の共有メモリ並列計算機システムの形態における実数データの整数部分に基づく分類処理を説明するためのフローチャートである。図 1 7 では、1 から n までのインデックスを 2 つに分割して、それぞれを 2 つのプロセッサ 1 2 により並列に処理している。

【 0 1 3 1 】

ここで、ステップ 1 7 0 4 とステップ 1 7 0 8 における計算処理においては、先に説明したベクトル計算機の場合と同様に、カウンタ c o u n t の更新処理と、実数データの分類先を l i s t に記録する処理とを、図 3 の付帯演算機 3 0 を用いて処理するのである。

【 0 1 3 2 】

以上説明されたように、ベクトル計算機システム及び共有メモリ計算機システムによる各実施の形態の本発明の計算機システムは、メモリバンク 4 0 に接続された付帯演算機 3 0 において、データの更新処理やカウンタの加算処理を行うことにより、正確かつ高速に演算を処理することができる。この際、各メモリバンク 4 0 におけるデータの更新は逐次に実行されるが、それぞれのメモリバンクが他のメモリバンクと独立に動作するため、全体として高速な演算が行われる。

【 0 1 3 3 】

また、上述された本発明の計算機システムの、ベクトル計算機システムによる形態と、共有メモリ計算機システムによる形態とは、合わせて実施することができる。つまり、図 2 に示される共有メモリ計算機システムの各プロセッサに、本発明のベクトル演算を実行するベクトルプロセッサを用いることが可能である。

【 0 1 3 4 】

以上好ましい実施の形態及び実施例をあげて本発明を説明したが、本発明は必ずしも上記実施の形態及び実施例に限定されるものではなく、その技術的思想の範囲内において様々に変形して実施することができる。

【 0 1 3 5 】

【発明の効果】

以上説明したように本発明の計算機システムによれば、以下のような効果が達成される。

【 0 1 3 6 】

第 1 に、本発明の計算機システムでは、複数のメモリバンクのそれぞれに独立かつ並列に動作する付帯演算機を備えて、各メモリバンク毎に局所的な排他処理を行うことにより、ベクトル演算や並列処理における個々のデータの一貫性を保った更新処理を、正確に容易に高速に処理することができる。

【 0 1 3 7 】

第 2 に、本発明の計算機システムにより、メモリ内のデータの更新処理や、カウンタの更新を伴うデータの分類処理等の、従来ではベクトル化（又、並列化）が困難であった各種の処理を、容易にベクトル化（又、並列化）して高速に処理することができる。

【図面の簡単な説明】

【図 1】 本発明の計算機システムをベクトル計算機システムに適応した実施の形態の一実施例の構成を示すブロック図である。

【図 2】 本発明の計算機システムを並列計算機システムに適応した実施の形態の一実施例の構成を示すブロック図である。

【図 3】 本発明の付帯演算機 3 0 とメモリバンク 4 0 との接続と、付帯演算機 3 0 の一実施例の構成を示すブロック図である。

【図 4】 本発明の計算機システムの処理を説明するための図であり、

【図 5】 本発明の計算機システムの処理を説明するためのフローチャートである。

【図 6】 本発明の計算機システムによるカウンタの加算処理を説明するためのフローチャートである。

【図 7】 行列演算の一例を示す図である。

【図 8】 “ $y = y + A x$ ” の行列演算を処理するアルゴリズムの一例を示すフローチャートである。

【図 9】 図 8 に示されるアルゴリズムを用いた、図 7 の “ $y = y + A x$ ” の行列演算を示す図である。

【図 1 0】 図 8 に示されるアルゴリズムを用いた、図 7 の “ $y = y + A x$ ” の行列演算を示す図である。

【図 1 1】 本発明の図 1 のベクトル計算機システムの形態における行列演算 “ $y = y + A x$ ” の処理を説明するためのフローチャートである。

【図 1 2】 本発明の図 2 の共有メモリ並列計算機システムの形態における行列演算 “ $y = y + A x$ ” の処理を説明するためのフローチャートである。

【図 1 3】 実数値データの整数部分に基づく分類処理を説明するための図である。

【図 1 4】 実数データの整数部分に基づく分類を処理するアルゴリズムの一例を示すフローチャートである。

【図 1 5】 図 1 4 に示されるアルゴリズムを用いた、図 1 3 の例の実数データを分類する処理を説明するための図である。

【図 1 6】 本発明の図 1 のベクトル計算機システムの形態における実数データの整数部分に基づく分類処理を説明するためのフローチャートである。

【図 1 7】 本発明の図 2 の共有メモリ並列計算機システムの形態における実数データの整数部分に基づく分類処理を説明するためのフローチャートである。

【符号の説明】

- 1 1 ベクトルプロセッサ
- 1 2 プロセッサ
- 2 0 メモリ制御部
- 3 0 付帯演算機
- 3 1 付帯演算機制御部
- 3 2 付帯演算機演算部
- 3 2 - 1 スイッチ
- 3 3 データ用シフトレジスタ
- 3 4 アドレス用シフトレジスタ
- 3 4 - 1 スイッチ
- 3 5、3 6、3 7、3 8 レジスタ
- 4 0 メモリバンク
- 6 0 アクセス調停部

7 1 メモリ制御線

7 2 データ線

7 3 アドレス線

7 4、7 5 アクセス調停線

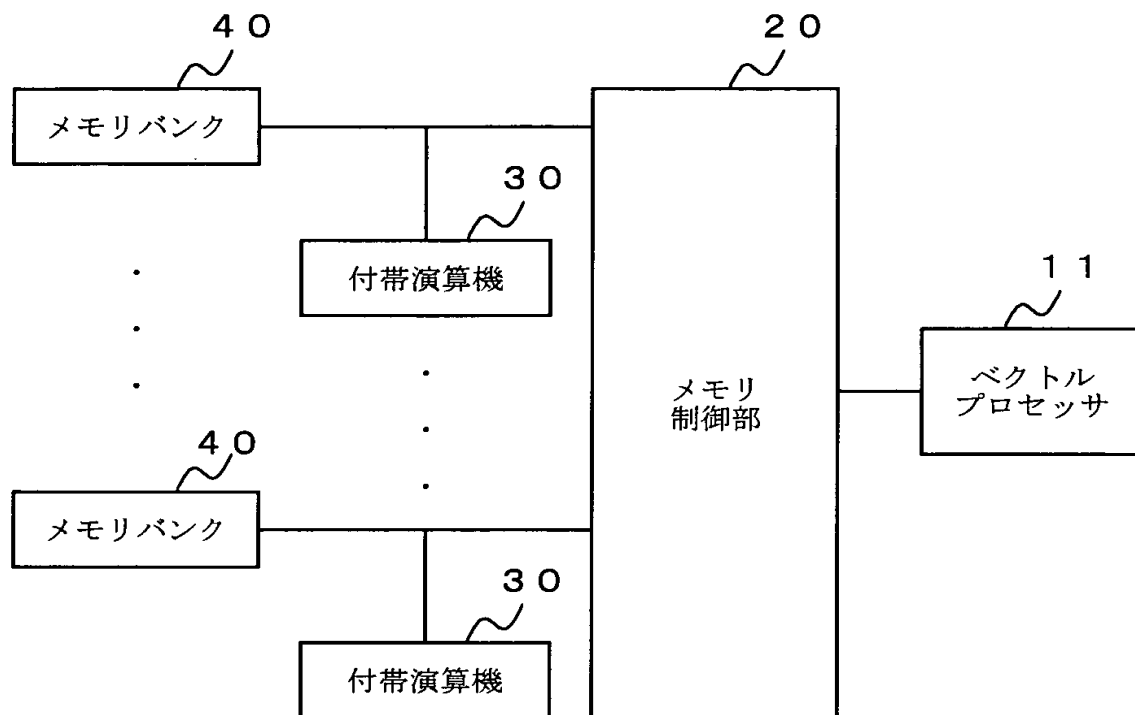
7 6 付帯演算機制御線

7 7 データ線

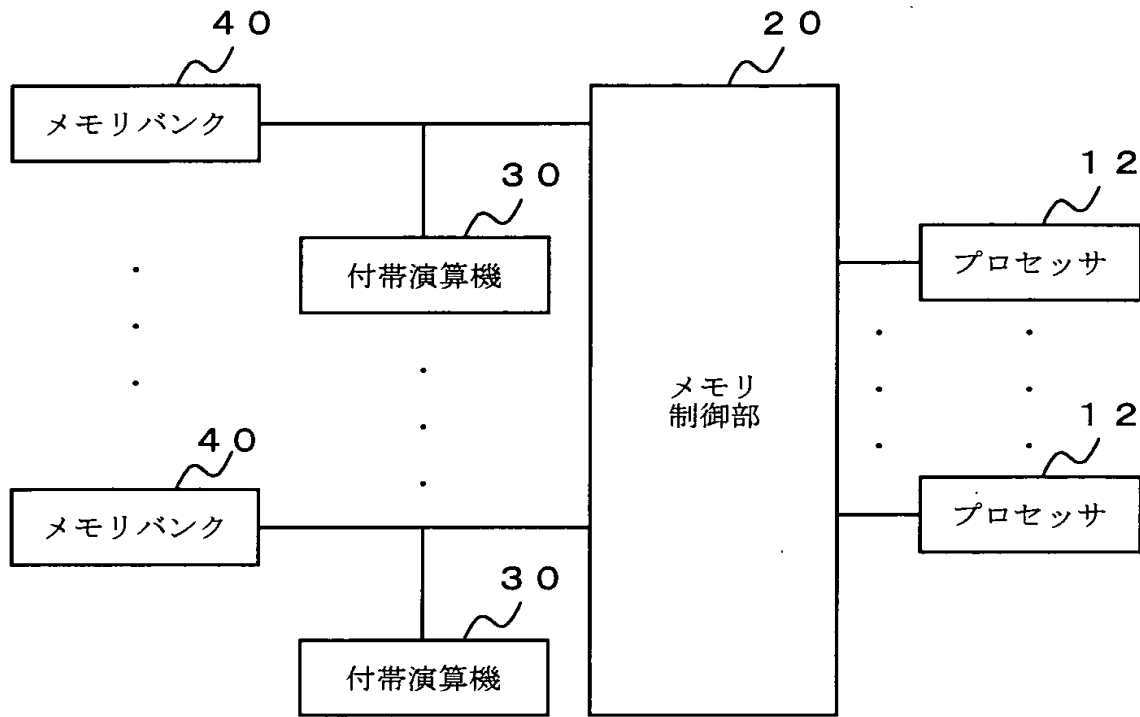
8 1、8 2、8 3、8 4、8 5、8 6 スイッチ

【書類名】 図面

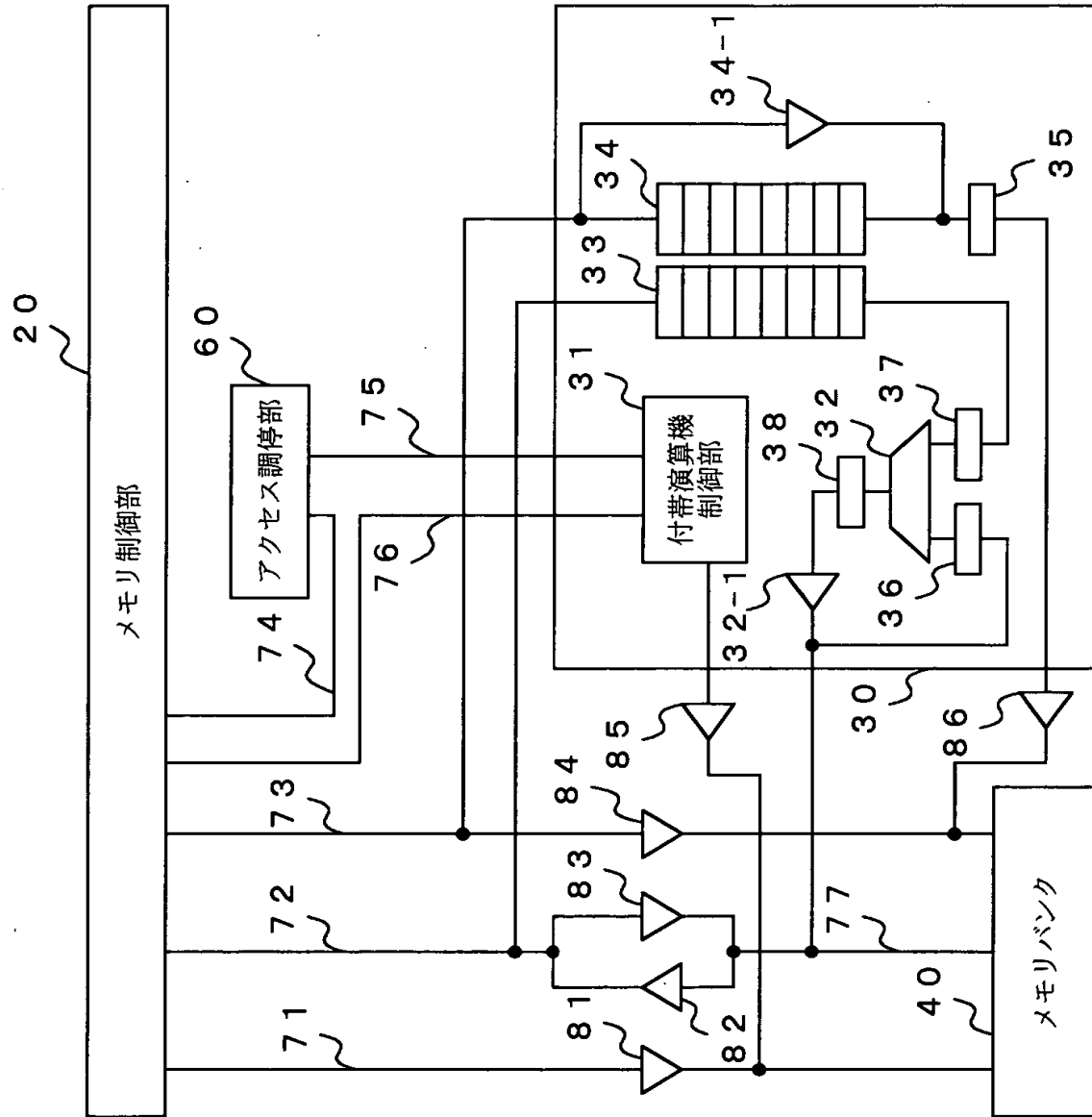
【図 1】



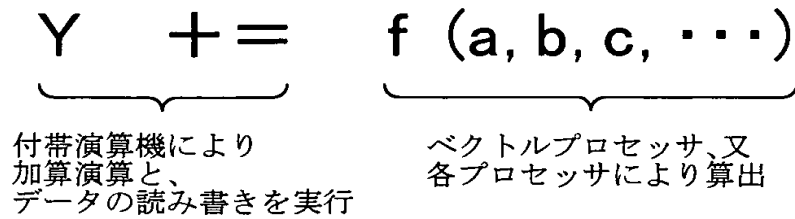
【図 2】



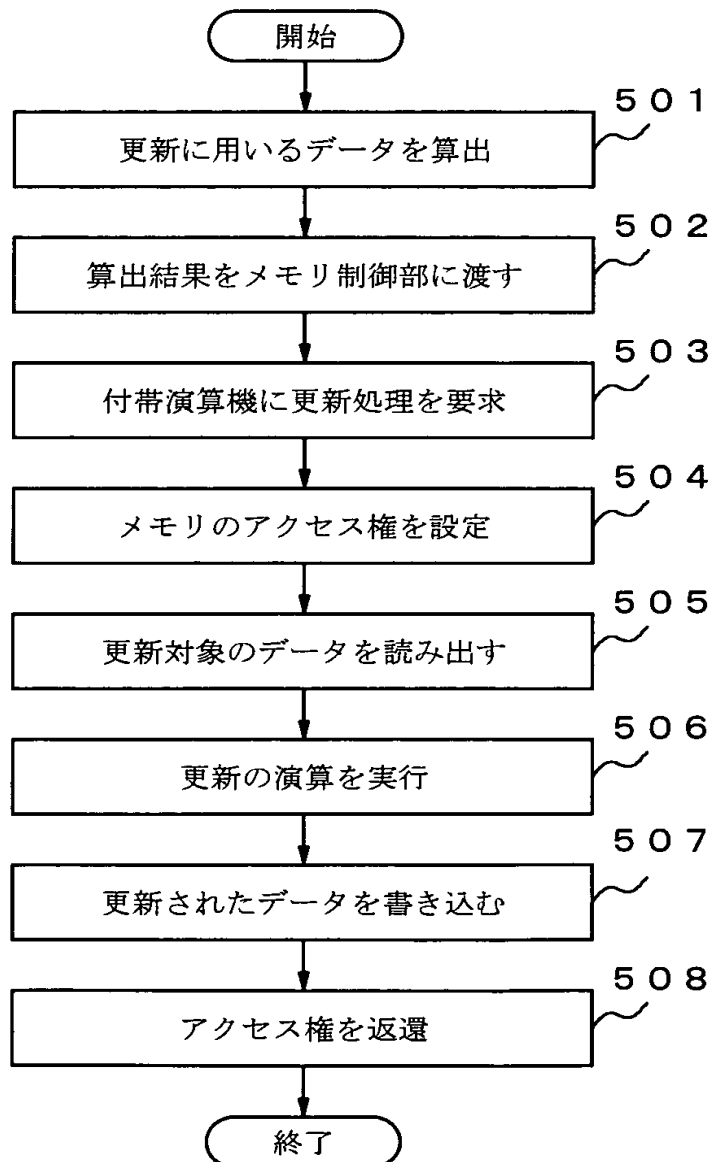
【図 3】



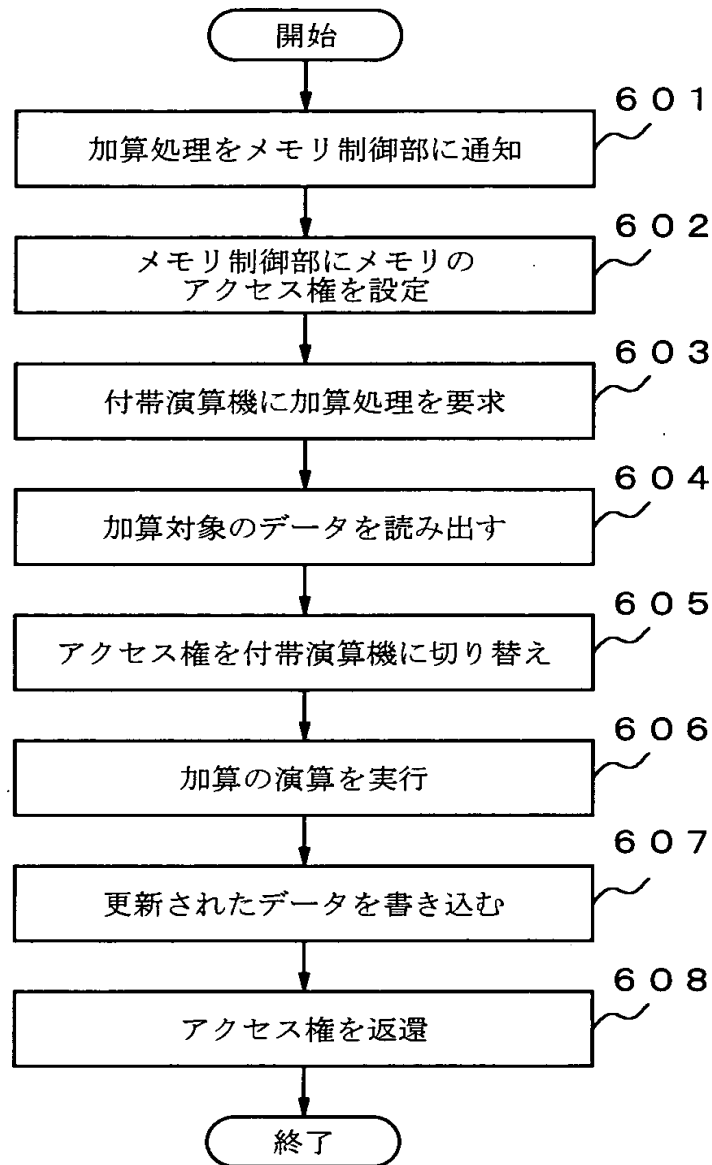
【図 4】



【図 5】



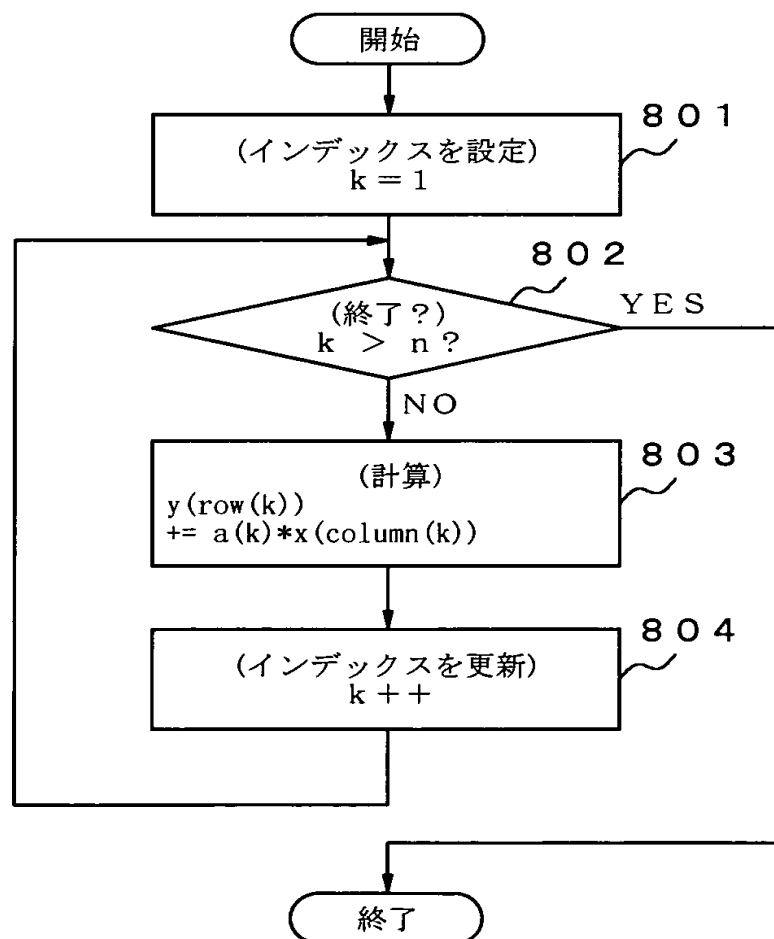
【図 6】



【図 7】

$$y = \begin{pmatrix} 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{pmatrix} + \begin{bmatrix} 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix} \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

【図 8】



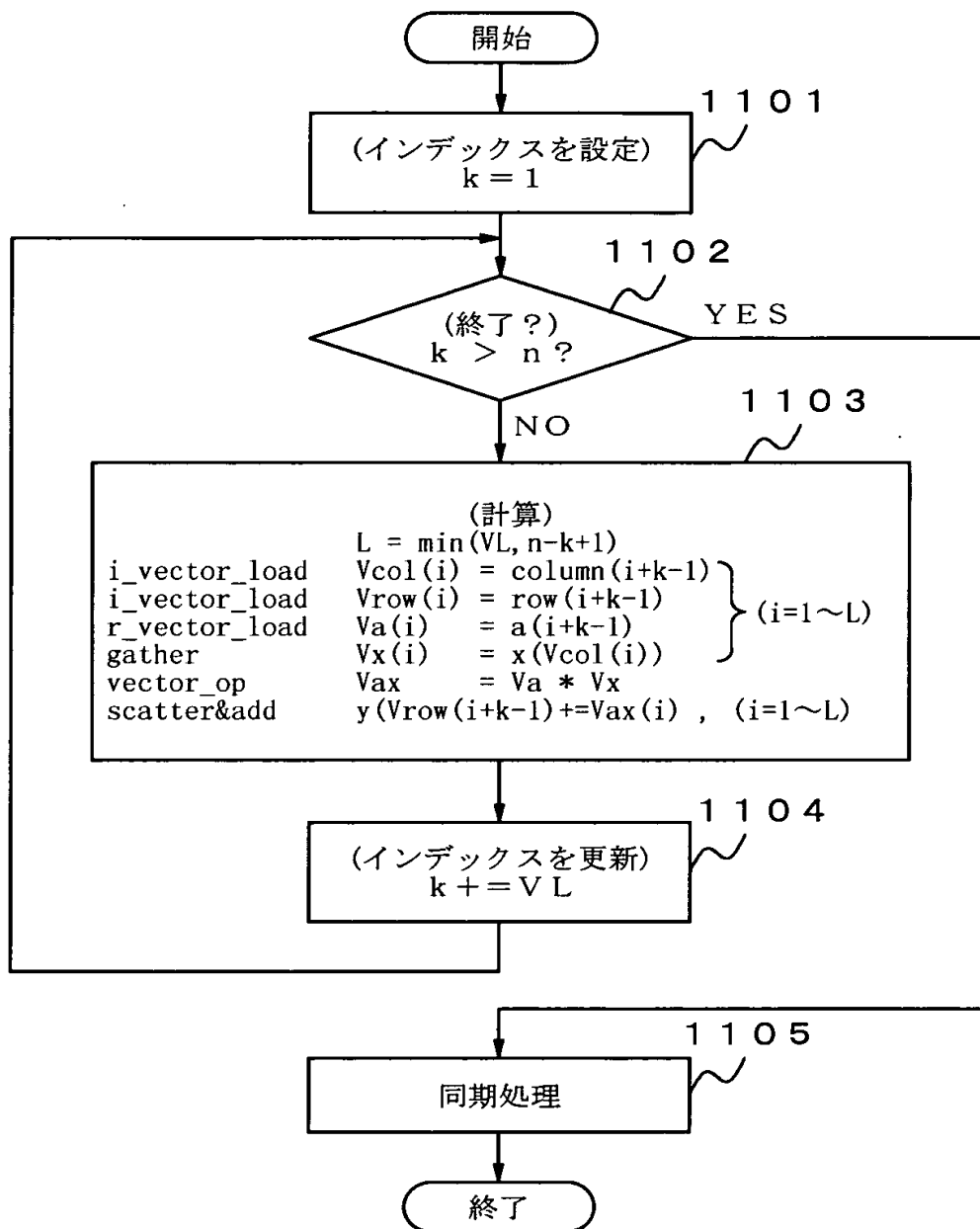
【図 9】

k	a(k)	column(k)	y(column(k))	a(k)*x(column(k))
1	6	1	2	1 2
2	5	2	0	0
3	4	1	2	8
4	3	5	1	3

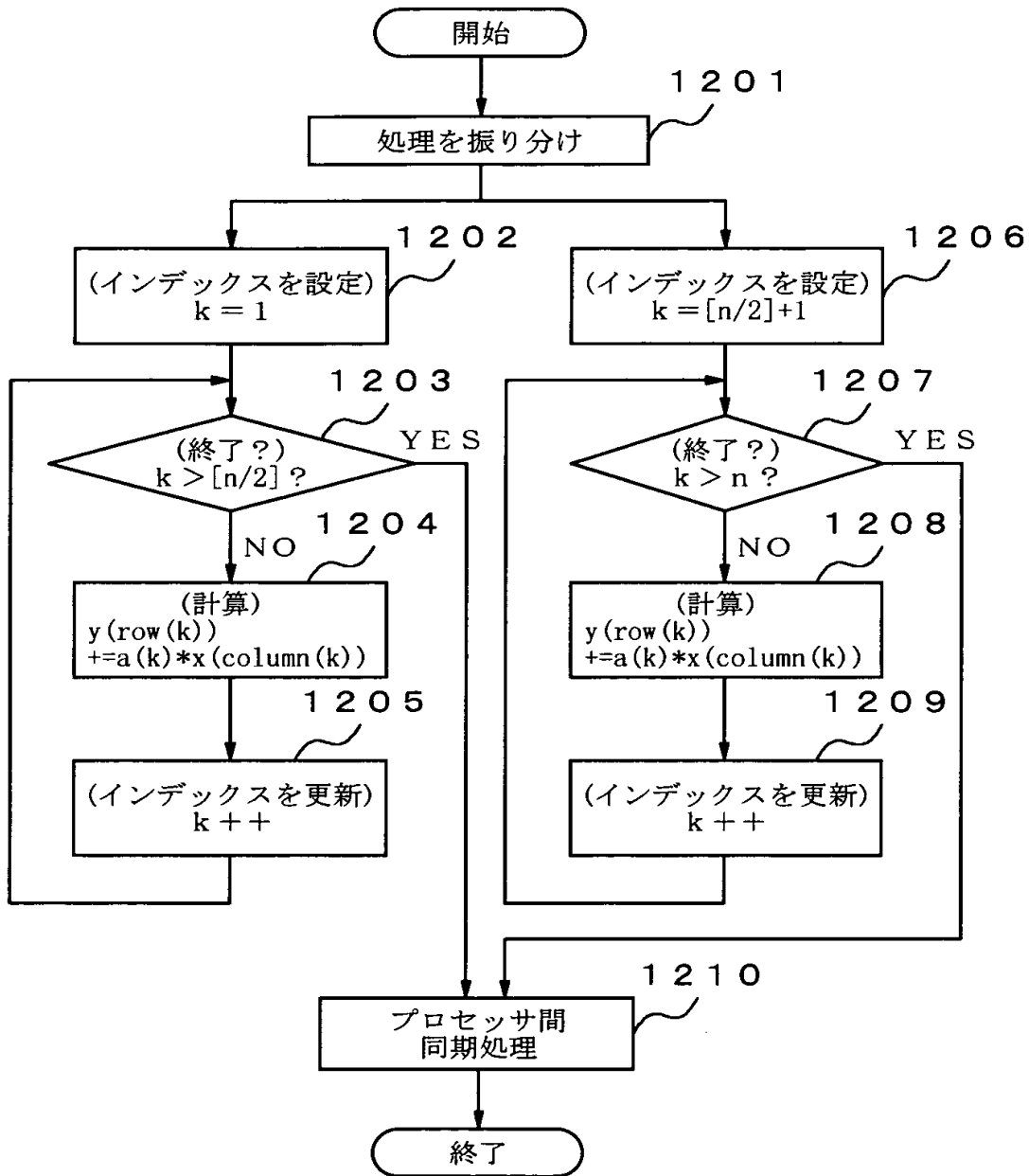
【図 1 0】

k	a(k)*(column(k))	row(k)	y(row(k))	y(row(k))+a(k)*x(column(k))
1	1 2	1	7	1 9
2	0	1	1 9	1 9
3	8	3	9	1 7
4	3	5	1 1	1 4

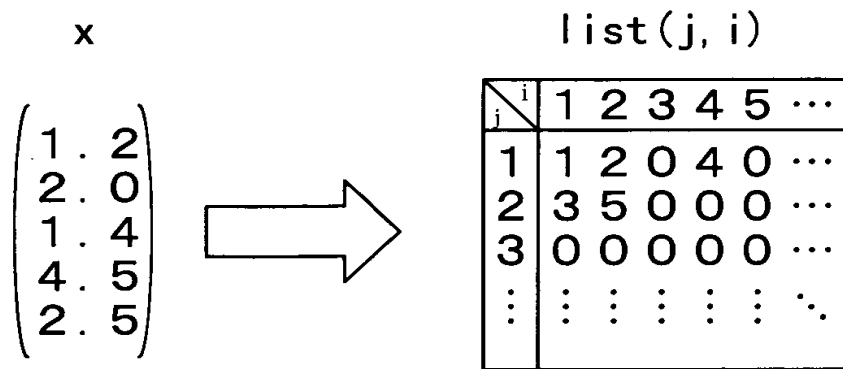
【図 11】



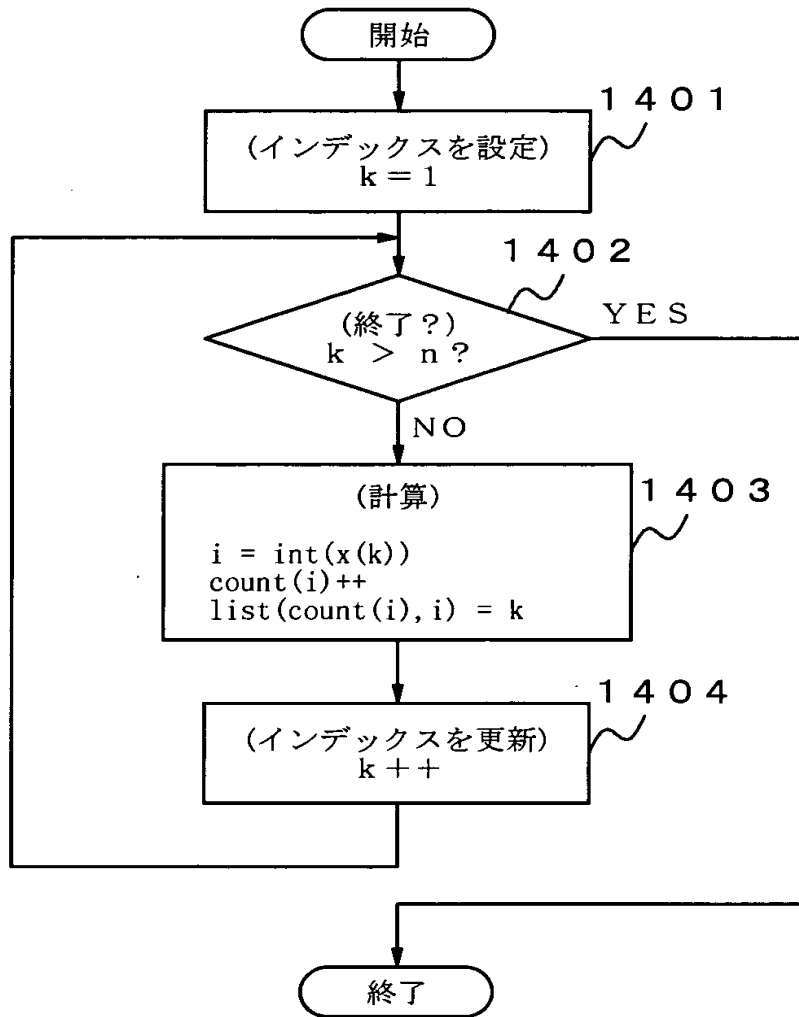
【図 12】



【図 1 3】



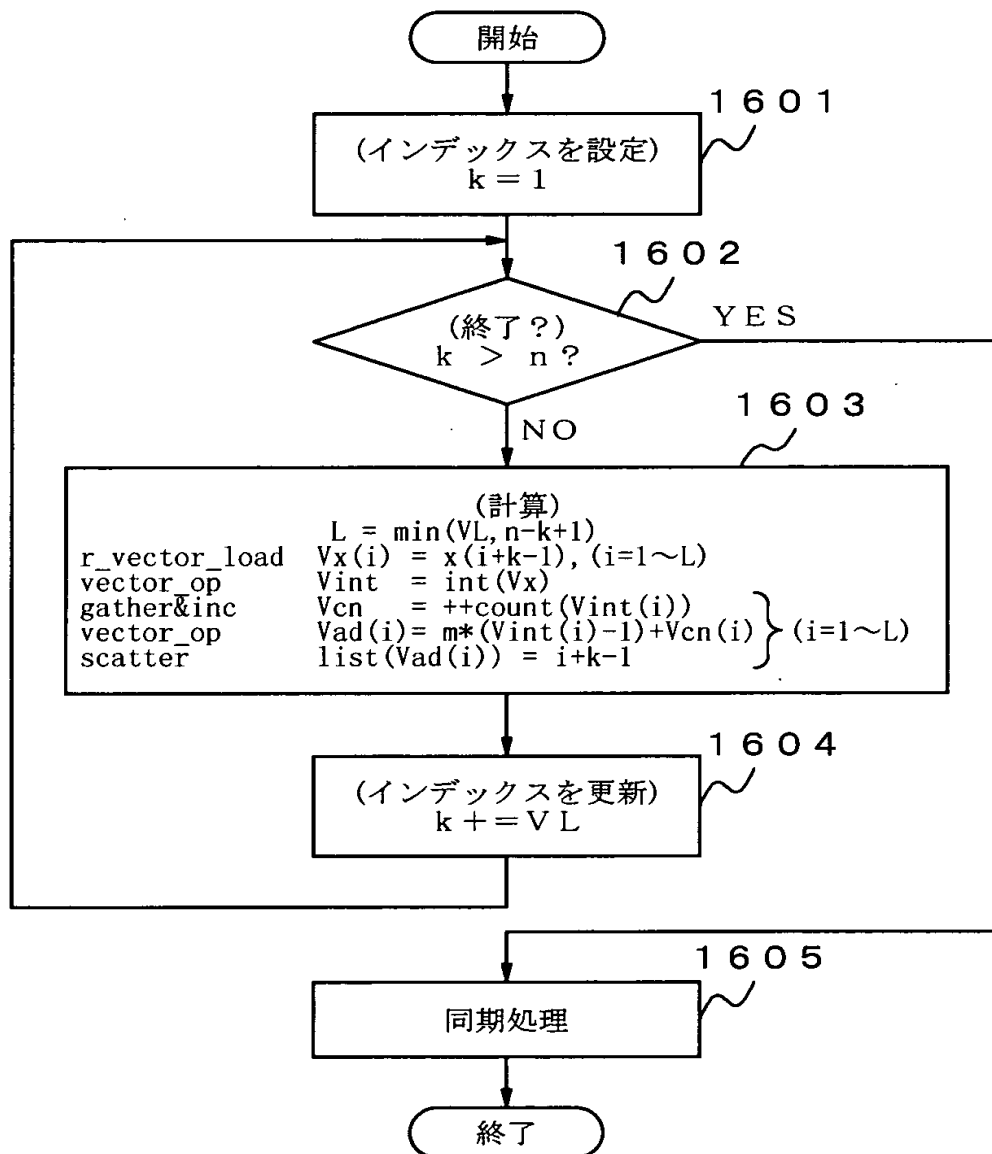
【図 1 4】



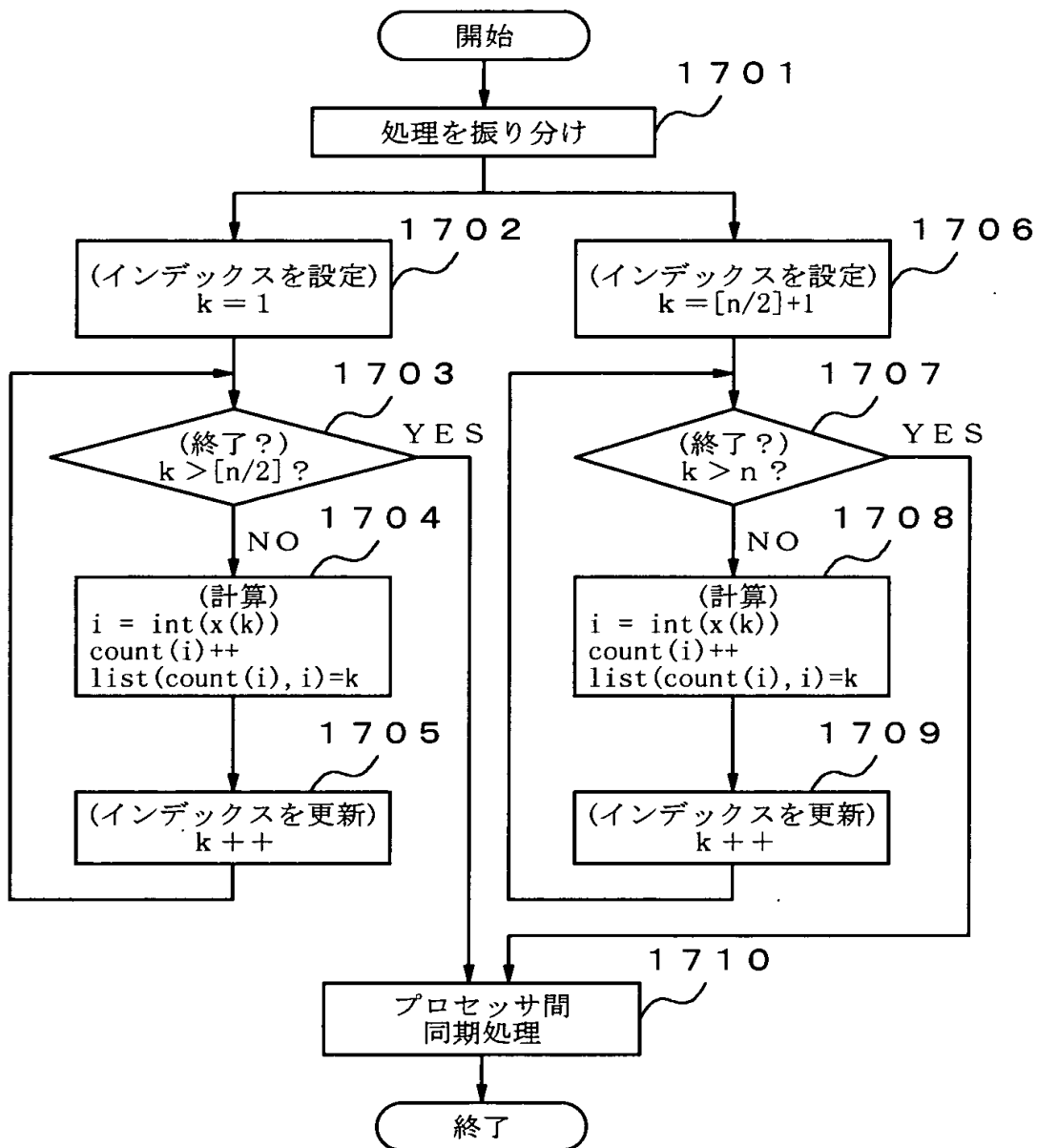
【図 1 5】

k	x(k)	int(x(k))	座標	count(i)					i / ステップ
				1	2	3	4	5	
1	1.2	1	(1, 1)	1	0	0	0	0	1
2	2.0	2	(1, 2)	1	1	0	0	0	2
3	1.4	1	(2, 1)	2	1	0	0	0	3
4	4.5	4	(1, 4)	2	1	0	1	0	4
5	2.5	2	(2, 2)	2	2	0	1	0	5
				2	2	0	1	0	

【図16】



【図17】



【書類名】 要約書

【要約】

【課題】 データの更新処理や、カウンタの更新を伴うデータの分類処理等の各種の処理を、容易にベクトル化して高速に処理することのできる計算機システムを提供する。

【解決手段】 複数のメモリバンク 4 0 を備える計算機システムにおいて、計算処理を制御するベクトルプロセッサ 1 1 からの制御を受けて指定された演算をベクトルプロセッサ 1 1 から独立に処理する付帯演算機 3 0 を各メモリバンク 4 0 毎に備え、各付帯演算機 3 0 は、対応するメモリバンク 4 0 内に記録されたベクトルプロセッサ 1 1 により指定されたアドレスのデータを読み出し、読み出したデータに対してベクトルプロセッサ 1 1 により指定された演算を実行し、演算結果のデータを指定されたアドレスに書き込むことにより、当該アドレスのデータの更新処理を行うことを特徴とする。

【選択図】 図 1

特 2000-299683

認定・付加情報

特許出願の番号	特願2000-299683
受付番号	50001266929
書類名	特許願
担当官	第七担当上席 0096
作成日	平成12年10月16日

<認定情報・付加情報>

【提出日】	平成12年 9月29日
-------	-------------

次頁無

出 願 人 履 歴 情 報

識別番号 [000004237]

1. 変更年月日	1990年 8月29日
[変更理由]	新規登録
住 所	東京都港区芝五丁目7番1号
氏 名	日本電気株式会社